Write a Spotlight Plug-in Today

Hands-On Session 101

These are confidential sessions—please refrain from filming or taking pictures.

Application Technologies

Spotlight Importers

Dominic Giampaolo
Spotlight Weenie

http://developer.apple.com/wwdc2005/

Agenda

- What is Spotlight?
- Importer Background
- Writing an Importer
- Importer Tips & Tricks
- Advanced Importers

Spotlight Architecture

http://developer.apple.com/wwdc2005/

What Is Spotlight?

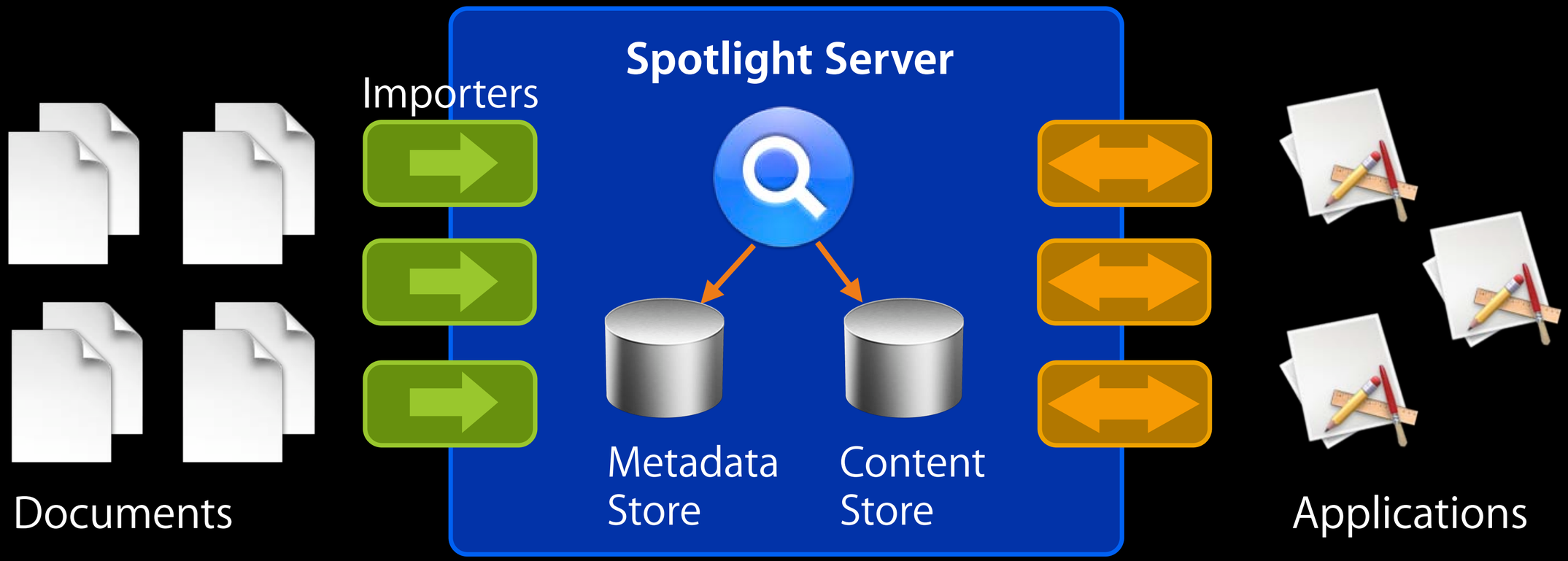
- A system for storing, querying and retrieving attributes about files
- The Spotlight system is made up of:
  - A server
  - Daemons that help the server
  - Importers that publish metadata
  - Client API (part of CoreServices)

Why Do You Care?

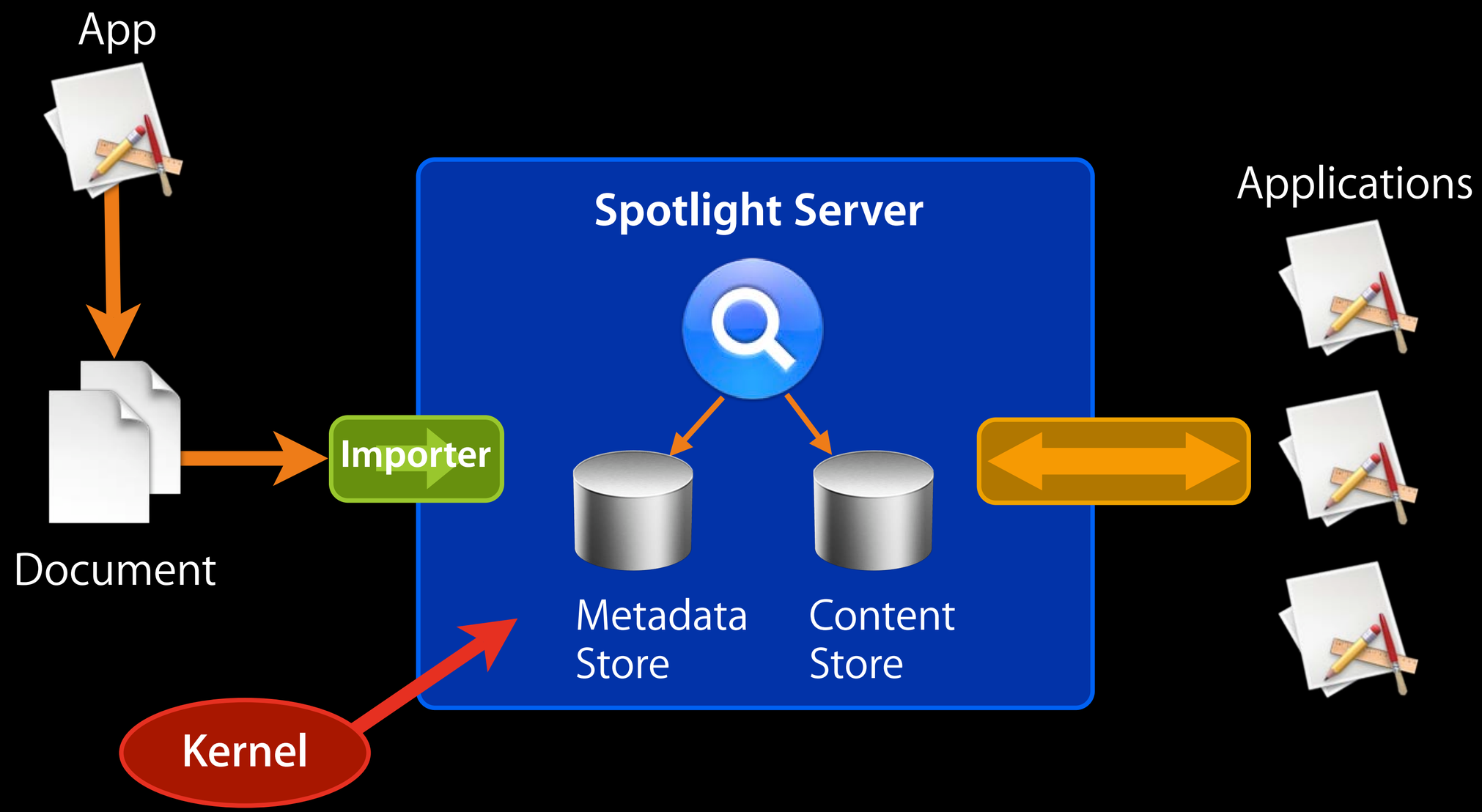
Because Spotlight helps…

- Makes your documents easier to find
- Makes your documents display more richly
- Requires no code changes in your app
- Easy to share data with other apps

Spotlight Architecture



Spotlight Architecture



Spotlight Concepts

- MDImporter
  - Importers are plug-ins loaded by Spotlight
  - Importers publish metadata from a file
  - An importer handles a set of file formats (UTI types)
- MDItem
  - Items represent files
  - Items are made up of attributes
  - Attributes are name/type/value
- MDQuery
  - A way to find Items

Ways to Integrate with Spotlight

- Put useful metadata in your documents (and preserve it)
- Write an importer
- Other things you can do to help:
  - Don‘t type files incorrectly!
  - Put your temp files where Spotlight won’t index
    - /tmp
    - A directory whose name ends in `.noindex` or `.build`

Demo

http://developer.apple.com/wwdc2005/

Apple Importers

- Images (JPEG, PNG, TIFF, GIF, DNG, PSD, BMP, Raw, etc)
- Audio (MP3, AAC, AIFF, AIFC, Digi SD2, etc)
- QuickTime
- PDF
- RichText (RTF, RTFD, Word)
- AppleWorks, Keynote, Pages, Excel, Powerpoint
- iCal, AddressBook, iChat logs, Mail

Importers

http://developer.apple.com/wwdc2005/

Importers

- Rules of the game
- A simple Importer
- Performance
- Tips, Tricks and Debugging
- An advanced Importer

Rules of the Game

- Importers should publish metadata that:
  - Facilitates user searches
  - Allow for richer previews
- Importers should not use Spotlight as:
  - A DB for private data
  - Icon previews, large blobs
- Too much “noise” can confuse the user
- No user-interface!

Rules of the Game: UTI Types

- UTI = Uniform Type Identification
- The filename extension or file Type Code determine the UTI type
- For any UTI type there is only one importer
- One importer can handle multiple UTI types
- You have to get the UTI declarations right

Rules of the Game: Attributes

- Examples of good attributes:
  - Copyright, title, author, dimensions
  - Strings the user entered as labels or names
  - Names that the user might associate with a document
- Text content is a special attribute
  - `kMDItemTextContent`
- Examples of bad attributes:
  - App-specific implementation details
  - Binary data

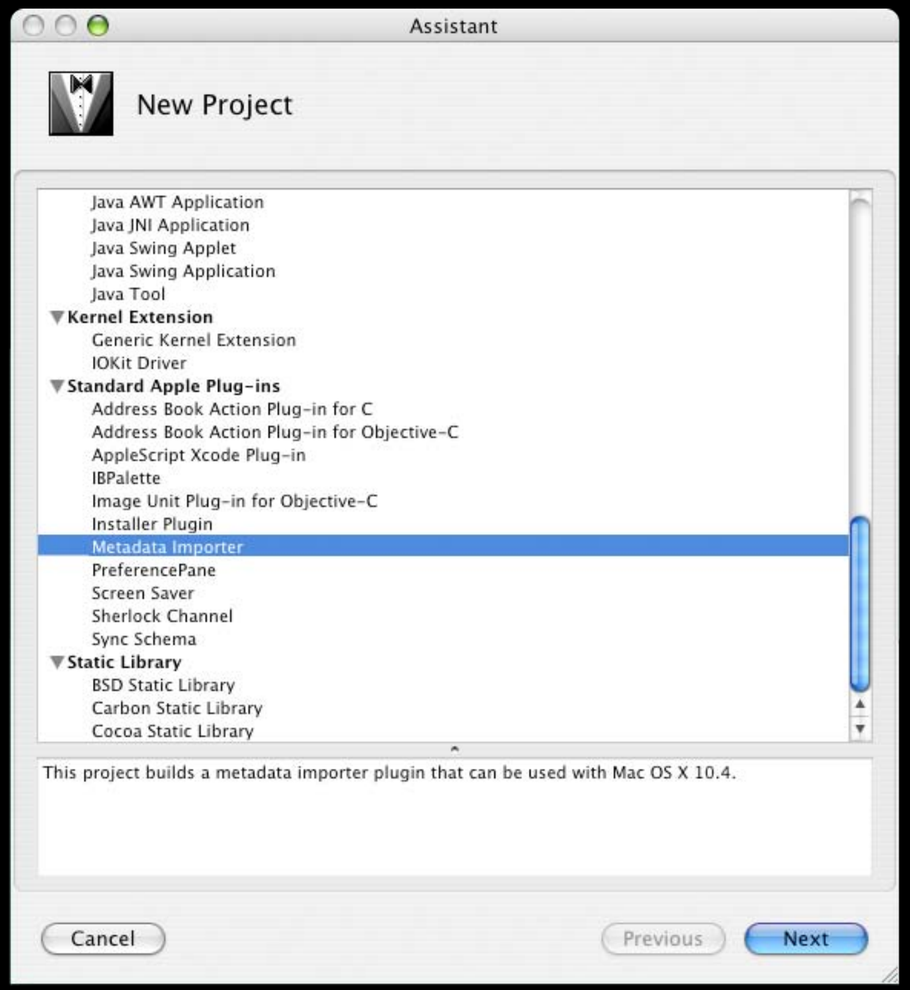
Rules of the Game: Attributes

- General pre-defined attributes
  - `kMDItemTitle`, `kMDItemAuthors`, `kMDItemKeywords`, `kMDItemProjects`
- Images/Movies
  - `kMDItemPixelHeight/Width`, `kMDItemColorSpace`, `kMDItemAperture`
- Audio
  - `kMDItemDurationSeconds`, `kMDItemAlbum`, `kMDItemAudioBitRate`
- Full list of attributes
  - `MDItem.h` (in the `Metadata.framework` under `CoreServices.framework`)

Writing a Basic Importer

- Start with the Metadata Importer template in Xcode
- Edit the Info.plist
- Implement `GetMetadataForFile()`
- Install your Importer

Xcode Metadata Importer Template



Edit the Info.plist

- Keys to edit:
  - If you need to define the UTI, uncomment
    `<key>UTExportedTypeDeclarations</key>`
    - Make sure your UTI's are correct!
    - The `UTTypeConformsTo` section is especially important
  - Identify the UTI types the Importer handles
    `<key>LSItemContentTypes</key>`
  - Edit the other informational fields (bundle name, version, etc)
  - Don't global search-and-replace com.apple!

Implement the Code

- Implement the `GetMetadataForFile()` function
- Efficiently extract the attributes from the file
- Add the attributes to the `CFDictionary`

Install Your Importer

- Install locations
  - Stand-alone:
    - /Library/Spotlight
    - ~/Library/Spotlight—but this is per-user ONLY
  - With an application:
    - Foo.app/Contents/Library/Spotlight
- Precedence is:
  - ~/Library/Spotlight
  - /Library/Spotlight
  - /System/Library/Spotlight
  - App-bundled importers

A Simple Importer

http://developer.apple.com/wwdc2005/

A Simple Importer

- In the Info.plist
  - Define the UTI type
  - Bind the Importer to that UTI type
- Write the code
- Install It

Performance

- MDImporter’s run in several different contexts
- Check your performance with `mdimport -p`
- Be a good citizen (unlike the iff-importer)
- Avoid using lots of memory or cpu
- Use caution reading large files
- Don’t blindly re-use code

Tips for Importers

- Use standardized attribute names
- Avoid inventing new attribute names
- Don’t forget text content when applicable
- Don’t publish too much
- Remove attributes by not publishing one
- Delete `schema.xml` and `schema.strings` if not used

Fine Points of the Info.plist

- Use `UTImportedTypeDeclarations` instead of `UTExportedTypeDeclarations`
- Keep everything lower-case
- Don’t forget about the `OSType` attribute (if you need it)
- The `UTTypeConformsTo` section is crucial
  - Physical hierarchy (how the item is stored)
    - `public.data` for flat files
    - `com.apple.package` for package-based files
  - Semantic hierarchy (what is stored in the item)
    - `public.content`—generic document content
    - `public.composite-content`—mixed content document (RTFD, PDF)
    - `public.image, public.audio, etc`—specific content types

Command Line Tools

- Several tools can help
  - `mdimport`—Import files, print out global state
  - `mdls`—List the attributes of files
  - `mdfind`—Issue a query, see the results
- Read the man pages for more details on options

Testing and Debugging

- Start simple
  - Run `mdimport -L`
  - Check for your UTI with `mdimport -d1`
- Be careful if you have multiple copies
- Always start from a fresh Importer Template
- Check the types of the attributes you publish
- Debugging
  - `mdimport -d[1,2,3,4] path-to-import`
  - `gdb mdimport`
    - Set a breakpoint on `GetMetadataForFile()`
  - Set the project executable to be `mdimport` in Xcode

When It Still Doesn't Work…

- Pull out the heavy artillery:
  - `lsregister` can tell you everything
  - `lsregister -dump` will tell you who declares which types
    - Be sure to check the path of the bundle that declares a type!
  - `lsregister -f -v /path/to/importer`
  - The ultimate weapon: `lsregister -kill -seed`
- `lsregister` is part of the ApplicationServices framework:
  - `/System/Library/Frameworks/ApplicationServices.framework/Frameworks/LaunchServices.framework/Support/lsregister`

Advanced Importers

http://developer.apple.com/wwdc2005/

Defining New Attributes

- Edit the `<attributes>` section of the schema.xml file:
  - `<attribute name="com_foo_TrackTitle" type="CFString"/>`
  - `<attribute name="com_foo_UserID" type="CFNumber"/>`
- Edit the `<types>` section and…
  - Add each attribute your importer publishes to the `<allattrs>` list
    - If you want the attribute displayed, also add it to the `<displayattrs>` list
- Do not redefine existing attribute names
- Run `mdcheckschema` on your schema.xml file
- Run `mdimport -X` to display all the schemas
- Run `mdimport -A` to display all the schema strings

Attribute Names and Descriptions

- The schema.strings file (UTF-16 encoded) is where you
  - Define human-readable strings for your attributes:
    - `“com_foo_TrackTitle” = “Track Title”;`
  - Describe your attributes:
    - `“com_foo_TrackTitle.Description” = “The title of a track”;`
- Localization of attribute names in separate “.lprog” files:
  - For example, the Italian.lproj/schema.strings would be:
    - `“com_foo_TrackTitle” = “Traccia Titolo”;`
    - `“com_foo_TrackTitle.Description” = “Il titolo della traccia”;`

Modifiers of Attributes

- Modifiers for the `<attribute>` tag:
  - `multivalued=true`—allows for an array of values
  - `nosearch=true`—will not match in normal queries
  - `uniqued=true`—common values stored only once
- Examples:
  - `<attribute name="com_foo_TrackTitle" multivalued=true type="CFString"/>`
  - `<attribute name="com_foo_PrivateName" nosearch=true type="CFString"/>`
  - `<attribute name="com_foo_TeamName" uniqued=true type="CFString"/>`
- You can combine modifiers

Localized Attributes Values

- Create a CFDictionary with key/value pairs for each language code
- Store that dictionary as the value for your attribute name
- Example:
  - kMDItemDisplayName = {
      "" = "Address Book.app";
      da = "Adressebog";
      en = "Address Book";
      it = "Rubrica Indirizzi";
    };

Advanced Importers (the code)

http://developer.apple.com/wwdc2005/

Advanced Importer

- Defined new attributes
- Declared a human-readable form of the new attribute names
- Exported an array-valued attribute

Summary

- Spotlight uses published metadata to make documents
  - Easier to find
  - Display more richly
- Any file can be a part of Spotlight if…
  - It has an importer
- Importers publish metadata from files
- Importers are easy to write
- Be cautious with what you publish
- Performance is critical!

What You Need to Do

- If you have a custom file format…
  - *Write an importer!*
- Put useful metadata in your documents
- Preserve metadata in existing file formats
- Come visit us in the lab so we can help you

Related Sessions

Using Spotlight Query API's	Session 104 Tuesday 2:00pm
Spotlight Lab	Tuesday 3:30pm
Uniform Type Identifiers and Launch Services	Session 107 Tuesday 5:00pm
Spotlight Lab	Wednesday 9:00am

Documentation, Sample Code, and Other Resources

http://developer.apple.com/wwdc2005/

Q&A

Dominic Giampaolo
Software Engineer

http://developer.apple.com/wwdc2005/

WWDC2005

Worldwide Developers Conference